WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

ROUNDED BRANCH AND BOUND METHOD FOR
MIXED INTEGER NONLINEAR PROGRAMMING PROBLEMS

Jacob Akoka

WP 904-77   January 1977

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

ROUNDED BRANCH AND BOUND METHOD FOR

MIXED INTEGER NONLINEAR PROGRAMMING PROBLEMS

Jacob Akoka

WP 904-77   January 1977

ABSTRACT

Most of the research in the field of integer programming has been devoted to the case of integer linear programming.When one has to deal with non-lineraties, linearization techniques are used. But these techniques are not general enough to be applied to all the varieties of non-linearities. Besides, the techniques add more variables to the original problem, making large-scale problems very difficult to solve.

In this paper, a method is described to solve integer nonlinear programming problems . In this "bounded Branch and Bound" method, the arborescence has only N nodes, where N is the number of variables of the problem. Therefore, we store in the main memory of the computer only the informations relative to the N nodes. We can therefore solve large-scale integer nonlinear programming problems.

When the objective function and the constraints are convex, a procedure is given, which enables us to alleviate the arborescence. For non-convex problems, a specific method is described. While optimizing a problem of the arborescence, some "fortuitous integer variables" may appear. A procedure is described to schedule these variables. Our method is also specialized to the case of Boolean variables.

Five different criteria which enable us to chose the "separation variable" are described and ordered. Finally, a programming code was written in Fortran IV. We report the results obtained using this code. The CPU times necessary to solve the problems are very small. We can therefore expect to solve large-scale integer nonlinear programming problems in reasonable time.

INTRODUCTION

In this paper, we describe a method, called Bounded Branch and Bound method, which enables us to solve mixed (or pure) Integer Nonlinear Programming problems. The formulation of the problem is:

$$\text{Max} \quad \phi(x) \qquad x \varepsilon R^n$$

$$\text{Subject to}$$

$$h_\ell(x) \leq 0 \qquad \ell = 1, 2, \ldots, m \tag{c-1}$$

$$a_j \leq x_j \leq b_j \qquad j \varepsilon J = 1, 2, \ldots, n \tag{c-2}$$

$$x_j \quad \text{integer} \qquad j \varepsilon E \subseteq J \tag{c-3}$$

In (I-1), we give a brief outline of the Bounded Branch and Bound method.

A procedure to construct an oriented graph associated with the problems is described in (I-2). In (I-3), we develop a sepcific graphical language for the arborescence, and in (I-4) we describe 6 different rules which are to be used in the arborescence. Depending on the rule used, each node change from one state to another one. The transformation table is given in (I-5).

A detailed procedure describing the passage from node s to node t is given in (I-6). If the problem is convex (i.e. the objective function and the constraints are convex), two properties are developed. The first one allows us to refuse a node of the arborescence without investigation. The second one enables us to stop the optimization process of a problem before the end. These techniques are described in (I-7).

In (I-8), we relax the assumption that the problem is convex. Therefore, we develop a specific algorithm for the non-convex case. Some modifications are indicated in (I-9), in order to solve the case where all the variables are boolean.

At each node of the arborescence, we will have to solve a continuous nonlinear programming problem. When solving this problem some variables may become integer. Such variables are called "fortuitous integer variables" (fiV). A procedure to schedule these variable is developed in (I-10). At each node, we will have to choose a variable to become integer. Such variable is called the "separation variable". Five different criteria are given, in order to choose this variable (I-11). A proof of the convergence of the algorithm is given in (I-12) and in (I-13) we indicate the number of nodes to be stored in the main memory of the computer. In (I-14) we give a complete example.

In chapter II, we describe the numerical computations. Ten different problems have been solved using BBB and the five different criteria developed precedently (II-1). In (II-2), we describe three different methods which enable us to order the five criteria. In (II-3), we present a flowchart of the computer code that has been written. A very brief description of the most important modules is given.

In Appendix 1, we give the listings of the subroutines described in (II-3).

The Problem

Let us consider the following problem:

$$
(P) \begin{cases}
\text{Max} \quad \phi(x) \quad x \varepsilon R^n \\[1em]
\text{Subject to:} \\[1em]
h_\ell(x) \leq 0 \quad \ell = 1, 2, \ldots, m & \text{(c-1)} \\[1em]
a_j \leq x_j \leq b_j \quad j \varepsilon J = 1, \ldots, n & \text{(c-2)} \\[1em]
x_j \quad \text{integers} \quad j \varepsilon E \subset J & \text{(c-3)}
\end{cases}
$$

We assume that:

(i) $\phi(x)$ and $h_\ell(x)$ are nonlinear functions, continuously differentiable.

(ii) Constraints (C-1) define a domain which may be convex or non-convex

(iii) Constraints (C-2) define a parallelotope (Π).

   (Π) is assumed to be bounded.

(iv) Without loss of generality, we can assume that a_j and b_j are integers.

If $E \neq J$, only some of the variables must be integer. The problem to solve is said to be "mixed integer nonlinear programming" problems. In the following pages, we propose a method to solve problem (P).

I. The Bounded Branch and Bound Method (BBB)

I.1 Brief Outline of the Method

Let $S = \{ x\varepsilon R^n | x$ satisfies (c-1) and (c-2)$\}$

a) First, we solve the following problem:

$$(P_0) \quad \begin{cases} \text{Max} \quad \phi(x) \\ \text{S.t.} \\ \quad X \varepsilon S \end{cases}$$

Let $x°$ be the optimal solution of (P_0) (if it exists)

b) If $x°$ is integer: END

c) Otherwise, $\exists j_1 \varepsilon E | x°_{j_1}$ is not integer. We proceed to the separation of S into two subsets S_1 and S_2 such that:

$$S_1 = \{ x\varepsilon R^n | x \text{ satisfies (c-1) and (c-2)} \underline{\text{and}}$$

$$x_{j_1} \varepsilon (a_{j_1}, [x°_{j_1}]) \}$$

$$S_2 = \{ x\varepsilon R^n | x \text{ satisfies (c-1) and (c-2) and } x_{j_1} \varepsilon ([x°_{j_1}] + 1, b_{j_1}) \}$$

($[x°_{j_1}]$ means the integer value of $x°_{j_1}$). To S_1, we associate the following problem.

$$(P_1) \quad \begin{cases} \text{Max} \ \phi(x) \\ \text{s.t.} \\ x\varepsilon S_1 \end{cases}$$

To S_2, we associate problem P_2:

$$(P_2) \quad \begin{cases} \text{Max } \phi(x) \\ \text{s.t.} \\ x \varepsilon S_2 \end{cases}$$

Then, we solve one of the problems and put the other one in a list.

d) If all the $(x_i)_{i \varepsilon E}$ are integer, go to e . Otherwise go to c .

e) If this solution is better than the first one, store it. Otherwise,

refuse the node.

If there is a problem in the list, solve it and go to d. Otherwise: END

I.2   The Oriented Graph Associated with the Problem

Consider a set A such that $A \subseteq E$. Let $x_A = x_j |j \varepsilon A$ and $\bar{x}_A$ the integer

components of x_A satisfying (c-2). To $S = (A, \bar{X}_A)$ we associate the

following problem:

$$P(S) \quad \begin{cases} \text{Max } \phi(x) \\ \\ h_i(x) \leq 0 \qquad i \varepsilon I \\ \\ a_j \leq x_j \leq b_j \qquad j \varepsilon J \\ \\ x_j = \bar{x}_j \qquad j \varepsilon A \end{cases}$$

Let $\hat{x}_S$ and $\hat{\phi}_S$ be the optimal solution of P (S).

a) The couple $S = (A, \bar{X}_A)$ represents a node of the graph G.

b) To each node S, we associate its "level" in the graph, called t_S.

t_S is equal to the number of components of x which are integers.

c) If $A = \phi$ (we have only (c-1) and (c-2)), we solve the continuous nonlinear

programming problem. The correspondant node S_O is called the "root"

of the arborescence.

d) When $E(S) = E$, $\left(\text{where } E(S) = \{j \varepsilon E | x_j (S) \text{ is integer}\}\right)$ constraint (c-3) is

therefore satisfied. We have found a solution to problem (P).

The corresponding node is called "terminal node".

e) If S is not a "terminal node", let's consider the variable

$\beta \epsilon E - E(S)$ and let us define a successor T. The level of T is $t_T = t_S + 1$.

T is defined by the couple $(B, \tilde{x}_\beta)$ where:

$B = E(S) \cup \beta$

$\hat{x}_j = x_j, \forall \epsilon E(S)$

$\tilde{x}_\beta = [\hat{x}_\beta(S)]$ or $[\hat{x}_\beta] + 1$

f) In general, if $S = (A, \bar{X}_A)$ is not a terminal node, we consider the

nodes T (with level $t_S + 1$) defined by:

$\beta \epsilon E - E(S)$

$T = (B, \hat{x}_\beta)$

$B = E(S) \cup \beta$

$\tilde{x}_j = x_j, \forall_j \epsilon E(S)$

$\tilde{x}_\beta = [\hat{x}_\beta(S) + \gamma)$

i) If $\gamma = -1, 2, -3, \ldots$, the nodes T are in the left wing of the graph. The

origin of the left wing is the node corresponding to $\gamma = -1$

(ii) If $\gamma = 0, 1, 2, \ldots$, the nodes T are in the right wing. Its origin is

the node corresponding to $\gamma = 0$.

g) T is called the successor of S if ST is an arc of the arborescence.

T is called the descendant of S if an elementary path exists between S and T.

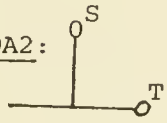## I-3   Different States of the Nodes in the Arborescence
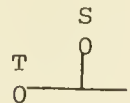
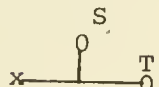We use the following graphical language:
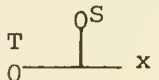
O   accepted node

x   refused node

R1: S_O is accepted but all its descendants are refused
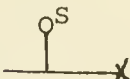
A2:   Node S_O is accepted.
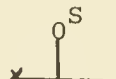
OA2:   T is a descendant of S. It is accepted. It belongs to the right wing originated from S. No nodes of the opposite wing were investigated.
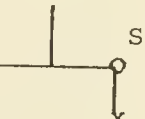
AO2:   Same as for OA2 but for a left wing
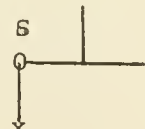
RA2:   The left wing is refused.

AR2:   The right wing is refused.

OR3:   S is accepted, the right wing is refused, the left wing not yet explored.

RO3:   S is accepted, the left wing is refused, the right wing not yet explored .

ORA3:   S is accepted. All its descendents are refused. Left wing not yet explored.

ARO3:   Same as ORA3 but the right wing is not yet explored.
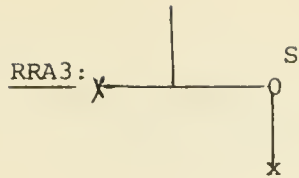
RRA3:   S is accepted, its descendants are refused.

The left wing is also refused.
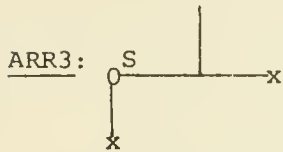
ARR3:   Same as RRA3 but the right wing is refused.
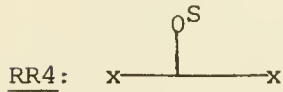
RR4:   S is accepted. Both wings originated from

S are refused. S, which has been previously

accepted, will now be refused because all its descendants

are refused.

I.4 RULES RELATED TO THE GRAPH G

We shall use the following rules in order to use the graph G.

<u>Rule 0</u>:　　If t=0, examine the node S_O . Therefore, we solve problem

(P) without constraint (C-3).　In order to do so, we use the

GRG (4) algorithm since COLVILLE (11) found it faster than

the other codes of nonlinear programming.　We let $\hat{\phi}=+\infty$

<u>Rule 1</u>:　　If the descendants of S_O are refused (this case is possible if

$\hat{\phi}_{S_-} \geq \hat{\phi}$), END of the exploration.　Then two possibilities exist:

we have the solution to problem (P) or the constraints (C-1),

(C-2) and (C-3) are incompatibles.

<u>Rule 2</u>:　　If at the level t, a node S is accepted and if its

descendants are not refused, examine a successor at the level t+1.

In this case, two possibilities exist:

(i) accept T　　　　if $\begin{cases} \hat{\phi}_T = +\infty \\ \\ T \text{ is not a terminal node} \\ \\ \hat{\phi}_T < \hat{\phi} \end{cases}$

(ii) refuse T

In all cases, add 1 to the current level.

<u>Rule 3</u>(i) If at a level t, we have one of the following possibilities,

examine the successor of T in its wing and accept or refuse it.

<u>Possibilities</u>

ORA3:　　　　T is accepted but all its descendants are
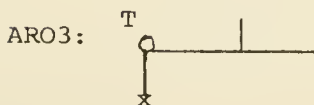
refuse.

ARO3:　　　　T is accepted but all its descendants are refused.

RRA3: T is accepted but all its descendants

are refused and the left wing is also
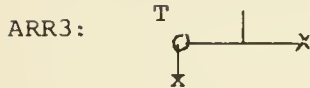
refused.

ARR3: T is accepted but all its descendants

are refused and the right wing is also

refused.

(ii) If at the level t, we have one of the following possibilities, examine

the origin of the opposite wing and accept or refuse it.

Possibilities

OR3: T is accepted, the right wing originated

from T is refused, left wing not yet
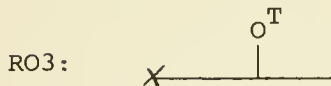
explored.

RO3: T is accepted, the left wing originated from

T is refused. The right wing not yet

explored.

Rule 4 When we move up in the arborescence (i.e. when t diminish), if we

have ARO3, ORA3 (respectively ARR3, RRA3) and if S corresponds

to an upper bound (respectively to a lower bound) of (P), refuse the

wing corresponding to S.

Rule 5 If we have the following possibility:

RR4: two wings refused.
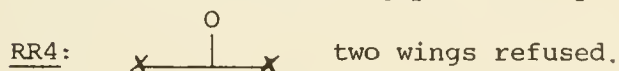
Refuse all the successors originated from both wings subtract 1

from current level.

Rule 6   If S is terminal (i.e. $\hat{x}_j(S)$ is integer $\forall_j \epsilon E$) then S is refused.

If $\hat{\phi}_S < \hat{\phi}$, then $\hat{x}(S)$ becomes the best solution.

If $\hat{\phi}_S \geq \hat{\phi}$, then the best solution is the current one.

I.5 Transformation of the States of the Nodes by the Rules

Using the rules described in (I.4), we obtain the following trans-
formation table:

State	The level before	Rule Applied	States after	The level after
R_1	0	1	end of the exploration	
A_2	t	2	OA2, AO2, OR3 RO3	t+1
OA2	t	2	OA2, AO2, OR3, RO3	t+1
RA2	t	2	OA2, AO2, OR3, RO3	t+1
AO2	t	2	OA2, AO2, OR3, RO3	t+1
AR2	t	2	OA2, AO2, OR3, RO3	t+1
OR3	t	3	AR2, RR4	t
RO3	t	3	RA2, RR4	t
ORA3	t	3	OA2, OR3	t
ARO3	t	3	AO2, RO3	t
RRA3	t	3	RA2, RR4 ,	t
ARR3	t	3	AR2, RR4	t

I-6.   PASSAGE FROM NODE S TO NODE T

Let S be an accepted node at the level t.   The solution of problem P(S) gave us $\hat{x}(S)$ and $\hat{\phi}_S$.   Besides we have $t = $ card $[E(S)]$. The node T, successor of S may be:

(i)   at the same level t as S.   In this case, it may be a successor in the same wing or at the origin of the opposite wing of S.

(ii) at the level t+1.   In this case, T is at the origin of a wing. The other wing is still unexplored if we move down in the arboresence, or, closed if we move up in the arborescence.

In order to solve P(T), we can choose one of the following strategies:

(a) If the integer value to try is $[\hat{x}_\beta]$

We solve the following auxiliary problem:

$$
P_i \quad
\begin{cases}
\text{Min } x_\beta \\[4pt]
\text{s.t.} \\[4pt]
h_i(x) \leq 0 & i = 1, \ldots, m \\[4pt]
a_j \leq x_j \leq 1_j & j = \epsilon\ J = \{1, 2, \ldots, n\} \\[4pt]
x_j = \bar{x}_j & \forall_j \epsilon\ E(S) \\[4pt]
[\hat{x}_\beta] \leq x_\beta
\end{cases}
$$

(b) If the integer value to try is $[\hat{x}_\beta] + 1$

In this case, we solve the following auxiliary problem:

$$
P_S \begin{cases}
\text{Max} \quad x_\beta \\[4pt]
\text{s.t.} \\[4pt]
h_i(x) \leq 0 & i = 1, \ldots, m \\[4pt]
a_j \leq x_j \leq l_j & j \in J = \{1, 2, \ldots, n\} \\[4pt]
x_j = \bar{x}_j & \forall_j \in E(S) \\[4pt]
x_\beta \leq [\hat{x}_\beta] + 1
\end{cases}
$$

Notes

- $\hat{x}(S)$ is a feasible point to the problems P_i and P_S.

- We can solve P_i and P_S using the same code (GRG) as the one used to solve P(S).

- Let $x^i = \{x^i_1, x^i_2, \ldots, x^i_n\}$ and $x^S = \{x^S_1, x^S_2, \ldots, x^S_n\}$ be the optimal solutions to P_i and P_S.

  (a) if $x^i_\beta > [\hat{x}_\beta]$ (respectively $x^S_\beta < [\hat{x}_\beta] + 1$) we can conclude that a solution that has $x_\beta = [\hat{x}_\beta]$ (respectively $[\hat{x}_\beta]+1$) will be infeasible for P(T). Therefore, we don't have to solve P(T). We can refuse node T without solving P(T).

  (b) if $x^i_\beta = [\hat{x}_\beta]$ (respectively $x^S_\beta = [\hat{x}_\beta]+1$) the solution x^i of P_i (respectively x^S of P_S) is infeasible for P(T).

I-7. Case where $\phi(x)$ and $h_i(x)$ are convex

Let's assume that the problem is convex and differentiable. These two properties will allow us:

(i) to refuse a node without solving its corresponding problem.

(ii) to interrupt the iterations in the process of optimization.

(A) Refusal of a node without investigation

Let T be an accepted node at the level t. Assume that T is in a wing originated by a node S at the level t-1. Let's call U the successor of T at this wing (Fig. 1).

Fig. 1

Let $S = (A, \bar{x}_A)$

$T = (B, x_\beta)$ $\left(B_1 \ \overset{\vee}{x}_\beta \right)$

where:

$B = A \cup \{\beta\}$ with $\beta \ \varepsilon \ E-E(S)$

$\tilde{x}_j = \overset{\vee}{x}_j = \bar{x}_j$, $\forall_j \ \varepsilon \ A$

$\tilde{x}_\beta = [\hat{x}_\beta(S)]+1$

$\overset{\vee}{x}_\beta = \tilde{x}_\beta + \Sigma$ with $\varepsilon = \begin{cases} +1 & \text{for a right wing} \\ -1 & \text{for a left wing} \end{cases}$

Let's write down Kuhn-Tucker conditions for problem P(T):

$\exists \ \lambda_i(T), \ \mu_k(T)$ such that

$$\lambda_i(T) \geq 0 \tag{4-a}$$

$$\mu_k(T) \geq 0 \quad \text{if} \quad \hat{x}_k(T) = a_k \tag{4-b}$$

$$\mu_k(T) \geq 0 \quad \text{if} \quad \hat{x}_k(T) = b_k \tag{4-c}$$

$$\mu_k(T) = 0 \quad \text{if} \quad a_k < \hat{x}_k(T) < b_k \tag{4-d}$$

$$\frac{\delta\phi}{\delta x_k} + \sum_{i=1}^{m} \lambda_i(T)\left(\frac{\delta h_i}{\delta x_k}\right)_{x=\hat{x}(T)} = \mu_k(T), \ k \quad J-B \tag{4-e}$$

$$\sum_{i=1}^{m} \lambda_i(T) \ h_i(\hat{x}(T)) = 0 \tag{4-f}$$

$\phi = \phi + \sum_{i=1}^{m} \lambda_i h_i$ is convex.   Therefore:

$$\phi[\hat{x}(U)] \geq \phi[\hat{x}(T)] + [\hat{x}(U) - \hat{x}(T)]\left(\frac{\delta\phi}{\partial x}\right)_{x \ = \ \hat{x}(T)}$$

$$\phi[\hat{x}(U) + \sum_{i=1}^{m} \lambda_i(T) \ h_i(\hat{x}(T)) \geq \phi[\hat{x}(T)] + \sum_{i=1}^{m} \lambda_i(T) h_i(\hat{x}(T))$$

$$+\sum_{K \varepsilon J-B}\left[\frac{\partial\phi}{\partial x_K} + \sum_{i=1}^{M} \lambda_i(T) \frac{\partial h_i}{\partial x_K}\right]_{x=\hat{x}(T)} [\hat{x}(U) - \hat{x}(T)]$$

$$+ \varepsilon\left[-\frac{\partial\phi}{\partial x_K} + \sum_{i=1}^{M} \lambda_i(T) \frac{\partial h_i}{\partial x_K}\right]_{x=\hat{x}(T)}, \tag{4-g}$$

Using (1-1) and (4-a), we have:

$$\sum_{i=1}^{\dot{m}} \lambda_i(T) \ h_i(\hat{x}(T)) \leq 0$$

Using (4-f), we obtain:

$$\sum_{i=1}^{m} \lambda_i(T) h_i(\hat{x}(T)) = 0$$

Using (4-b), (4-c), (4-d) and (4-e), we have:

$$\sum_{K \epsilon J-B} [\frac{\partial \phi}{\partial x_K} + \sum_{i=1}^{m} \lambda_i(T) \frac{\partial h_i}{\partial x_K}]_T [\hat{x}(U) - \hat{x}(T)] \geq 0$$

$$\rightarrow \phi[\hat{x}(U)] \geq \phi[\hat{x}(T) + \epsilon[-\frac{\partial \phi}{\partial x_\beta} + \sum_{i=1}^{M} \lambda_i(T) \frac{\partial h_i}{\partial x_\beta}]_{x=\hat{x}(T)} \qquad (4\text{-}h)$$

Let $\hat{\phi}_T = \phi(\hat{x}(T) + \epsilon[\frac{\partial \phi}{\partial x_\beta} + \sum_{i=1}^{M} \lambda_i(T) \frac{\partial h_i}{\partial x_\beta}] \quad x = \hat{x}(T) \qquad (4\text{-}i)$

and $\hat{\phi}_U = \phi[\hat{x}(U)] \qquad (4\text{-}j)$

Therefore, we obtain:

$$\hat{\phi}_T \leq \hat{\phi}_U \qquad (4\text{-}k)$$

Relationship (4-k) allows us to refuse node U without investigation (i.e. without optimizing). Indeed, let $\hat{\phi}$ be the value of the objective function corresponding to the best integer solution found till the precedent iteration.

If $\hat{\phi} \leq \hat{\phi}_T \Rightarrow \hat{\phi} \leq \hat{\phi}_U$

Therefore, we can refuse node U.

B. Interruption of the Iterations while Optimizing

(i) When solving problem P(S), at each iteration, the inequalities give us a lower bound of ϕ_S.

Let $\begin{cases} x^1 & \text{be the current variable at the iteration } 1, \\ \phi & \text{be the best integer solution found.} \\ \phi^1 & \text{be the lower bound associated with } x^1 \end{cases}$

when $1 \rightarrow \infty$, $\phi^1 \rightarrow \hat{\phi}_S$.

If node S must be refused because $\hat{\phi}_S > \hat{\phi}$, we can stop the iterations of P(S) when $\phi > \hat{\phi}$. We can obtain the lower bound using λ_i^1 and

μ_k^1 defined in the precedent paragraph.

(ii) When we solve problem P_i (respectively P_s), we try to make x_β equal to $[\hat{x}_\beta]$ (respectively $[\hat{x}_\beta]+1$).

- in P_i we minimixe x_β. If $\hat{x}_\beta$ could not be reached, we will know

  it when the lower bound associated to x_β will be greater than

  $[\hat{x}_\beta]+1$. We therefore add the following rule to our algorithm:

Rule 3bis: In moving up in the arborescence, if we are in

            ORA3 or RRA3 (respectively ARO3 or ARR3), then:

            (i) if we have an evaluation $\tilde{\phi}$ for T, the

                successor of S in its wing and

            (ii) if $\tilde{\phi} > \hat{\phi}$

            Refuse the wing corresponding to S.

If, while optimizing, we obtain for the problem P_i (respectively P_s) a solution close to x_β (i.e. $x_\beta \pm \varepsilon$), accept the node and stop the iterations.

I-8. Case where $\phi(x)$ and $h_i(x)$ are non-convex

In this paragraph, we don't assume convexity for functions $\phi(x)$ and $h_i(x)$. In this case, two cases of refusal of a node exist.

first case: Refuse any node which does not improve the objective function (i.e. when $\hat{\phi}_s > \hat{\phi}$). Therefore, refuse all its descendants.

second case: If a variable, already integer ,is at one of its bounds, refuse the successor corresponding to its wing.

Therefore, the rules to use are:

Rule 0: If t=0, investigate node S_o, (i.e., solve the continuous nonlinear programming) and let $\hat{\phi} = + \infty$.

Rule 1: If the descendants of S_o are refused, end of the exploration.

Rule 2: If at the level t, a node S is accepted and, if its descendants are not refused, investigate a successor T at the level t+1.

Rule 3: At the level t_m, examine in both wings, the successor U which has the smallest $|\gamma|$.

Using the "first case refusal" defined above, refuse or accept the descendance of S with a higher level.

Using the "second case refusal", refuse the successor of S in its wing.

· If T is the successor of another node in the same wing, refuse the latest node.

Rule 4: Moving up in the arborescence, if we have ARO3, ORA3 (respectively ARR3, RRA3) and if S corresponds to an upper bound (respectively to a lower bound), refuse the wing corresponding to S.

Rule 5: If we have RR4, refuse all the successors originated from both wings. Subtract 1 from the current level.

Rule 6: If S is terminal, then it is refused.

If $\hat{\phi}_S < \hat{\phi}$, then $\hat{x}(S)$ becomes the best solution

If $\hat{\phi}_S \geq \hat{\phi}$, then the best solution is the current one.

Globally, the main difference with the case where $\phi(x)$ and $h_i(x)$ are convex is Rule 3. Besides, note that every refusal is final and every acceptance is temporary.

1.9   Case where the Variables are Boolean

If all the variables are Boolean we have only the following
possibilities:
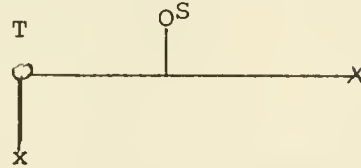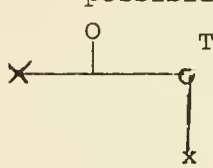
Use Rule 3 and
first case refusal
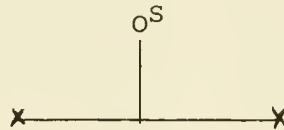
Use Rule 3 and
second case refusal
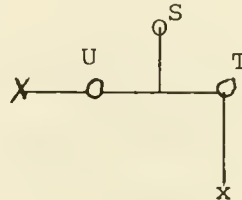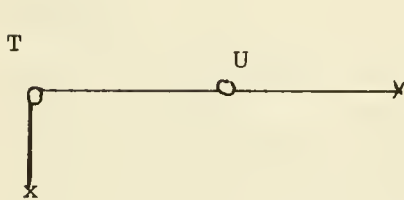
Use Rule 3 and
second case refusal

We can see that the method is converging whatever ϕ and h_i are.

But it is clear that the algorithm is effective if we know how to solve P(S).

Except for this restriction, all the results proposed preceedingly can

be applied without any hypothesis on ϕ and the domain defined by $h_i(x)$.

Assuming that ϕ is twice continuously differentiable ABADIE (5)(7)

showed how to solve P(S).

Function ϕ is made convex by letting

$$\bar{\phi}(x) = \phi(x) + \frac{1}{2} \; \alpha \sum_{j \in J} x_j (x_j - 1)$$

where α is a constant.

Let $H(x)$ and $\bar{H}(x)$ be the matrix of the second derivatives of ϕ and $\bar{\phi}$.

We have:

$$\bar{H}(x) = H(x) + \alpha I$$

If λ and $\bar{\lambda}$ are the smallest eigenvalue of the matrices $H*x)$ and $\bar{H}(x)$

we obtain the following relationship:

$$\bar{\lambda} = \lambda + \alpha$$

We therefore will have $\bar{\lambda} \geqslant 0$ for a big value of α. For this value of α, the function ϕ will be convex on the unit cube. By applying the same process to $h_i(x)$, we can obtain function $\bar{h}_i(x)$ convex.

THEOREM

Using the precedent fact, for a problem P which is twice continuously differentiable and totally bivalent, there exist a problem $\bar{P}$ such that the sub-jacent continuous problem is convex.

The latest theorem is in fact a generalization of HAMMER-RUBIN's (14) method for the case where ϕ is a function of the second degree and constraints are linear.

I.10 SCHEDULING OF THE FORTUITOUS INTEGER VARIABLES

By solving P(S), we obtain the optimal solution $(\hat{x}(S), \hat{\phi}_S)$.

Let $E(S) = \{j \notin E \mid \tilde{x}_j(S) \text{ is integer}\}$

We have $A \subset E(S) = E$.

If $E(S)-A \neq \phi$, we can say that by solving P(S), we obtained one or more "fortuitous integer variables".

By obtaining $\hat{x}(S)$ and $\hat{\phi}_S$ we have $\hat{x}_j = x_j$, $\forall_j \in A$. If by solving P(S) we have made appear $j_1 \in E-A$ such that:

$$\hat{x}_{j_1} = a_{j_1}$$

or

$$\hat{x}_{j_1} = b_{j_1}$$

The variable $\hat{x}_{j_1}$ is called "fortuitous integer variable". Let
$E_{f_S} = j_1, j_2 \ldots, j_f = $ set of the index of E corresponding to fortuitous integer variables in the solution x(S). We have:

$$E(S) \subset E$$

and

$$E(S) = E_{f_S} \subset A$$

If, while investigating the node S, the level was t, it becomes t+f after the analysis of x(S). The search for the fortuitous integer variables is made on the variables which are not integer at the level t. The order of discovering these variables is dependent on the order in which will be ranked the variables which are not yet integer. In order to eliminate many nodes from the arborescence, it will be interesting to have at the highest levels, the fortuitous integer variables which in its descendance there is a probability to

find a good integer solution. Therefore, we will have to change

the scheduling of those fortuitous integer variables.

CRITERION USED TO MODIFY THE SCHEDULING OF THE FORTUITOUS INTEGER VARIABLES (FIV)

We have indicated that the FIV are variables at their bounds. They

are therefore non-basic variables. In such a case, the components

corresponding to the reduced gradient are zero. Therefore, we

can use the following criterion:

"RANK THE FIV in the decreasing order of their reduced gradient

components". If the absolute value of a component of the reduced

gradient is small, a small augmentation of the value of the considered

point, will have no effect on the objective function. We can

expect that an integer solution found in the descendance of this

point will lead to a value of the objective function not too far

from the continuous optimal solution.

I .11 Choice of the Separation Variable

Let S be an accepted node at the level t. The solution of
P(S) gives us $\hat{x}(S)$, $\hat{\phi}_S$ and E (S). When we want to investigate a
successor T at the level t+1, we have to choose:

    (a)   an index $\beta\epsilon$ E-E(S)

    (b)   the wing originated from S and containing T.

We call E-E(S) or any subset of E-E(S), the "choice set".

In linear programming, we can compute for the variables included in

the choice set, penalties. Those penalties allow us to choice a

separation variable and a wing. In nonlinear programming, those

penalties are not applicable. Therefore, we propose other criteria.

Criterion 1 - choice of the closest variable to an integer

We first determine the variable x_j which is the closest to an

integer value, say e_j .

    (i)   if $e_j = [x_j]$ we solve problem R_{K+1} obtained by adding to P_K
          the constraint $x_j \leq e_j$.

    (ii) , if $e_j = [x_j] + 1$, we solve problem R'_{K+1} obtained by

adding to P_K the constraint $x_j \geq e_j$

$$
\begin{array}{ccc}
 & P_K & \\
R_{K+1} & & R'_{K+1} \\
x_j \leq e_j & & x_j \geq e_j
\end{array}
$$
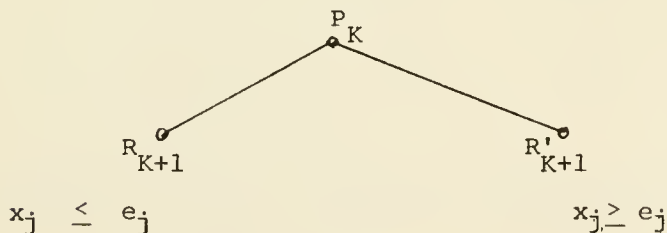
Criterion 2 - Choice of the Closest Variable to a Half-Integer

After solving P_K, each variable x_j is such that:

$$e_j < x_j < e_j + 1 \quad , \quad e_j \text{ integer} > 0$$

We choose x_j such that:

$$x_j = \frac{2e_j + 1}{2}$$

    (i)   if $e_j = [x_j]$, we solve problem R_{K+1} defined above

    (ii) if $e = [x_j] + 1$, we solve problem R'_{K+1} defined above.

Criterion 3 - Put in a List   the Problem Corresponding to the biggest psudo-cost
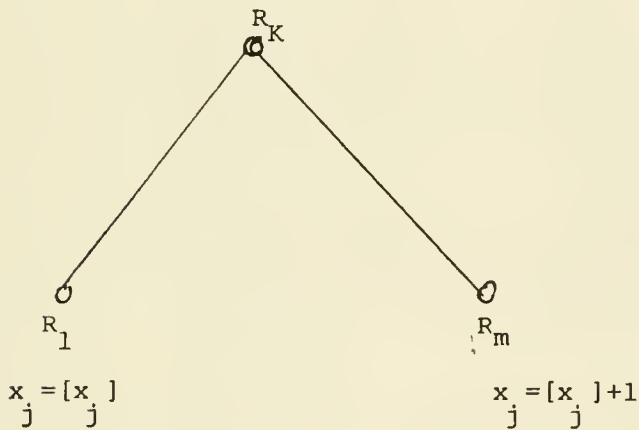
$$x_j = [x_j] \qquad\qquad\qquad x_j = [x_j] + 1$$

Let R_K be a node of the arborescence, and R_1 and R_m its direct successors obtained by adding respectively $x_j \leq [x_j]$ and $x_j \geq [x_j] + 1$

Define:

    ϕ_K = value of the objective function at node R_K

    ϕ_m = value of the objective function at node R_m

    f_j = the decimal part of x_j

We define the lower pseudo-cost relative to x_j, the   quantity

$$A_j(K) = \frac{\phi_K - \phi_1}{f_j}$$

and the upper pseudo-cost relative to x_j, the quantity

$$B_j(K) = \frac{\phi_K - \phi_m}{1 - f_j}$$

$A_j(K)$ is the diminution of the objective function   corresponding

   to a decrease of one unit of x_j

$B_j(K)$ is the diminution of the objective function corresponding to

   an increase of one unit of x_j .

The expression $\text{Max} [\text{Max} (A_j , B_j)]$ gives us an index i and
               j j
one pseudo-cost (either A_i or B_i).

> (i)   if the maximum is reached for A_i, put in the list the
>
>   problem obtained by adding to R_K the constraint
>
> $x_i \leq [x_i]$

> (ii)   If the maximum is reached for B_i, we put in the list the
>
>   problem obtained by adding to R_K, the constraint
>
> $x_i \geq [x_i]$.

In both cases, we solve immediately the problem which was not put

in the list.

Criteria 4 - Put in the list the problem corresponding to the Biggest Diminution of the Objective Function

This time, we use the expression

   $\text{Max} [\text{Max} (A_j f_j , B_j (1-f_j)]$
    j j

1-13 CONVERGENCE OF THE ALGORITHM

The number of nodes in the graph G is finite. Besides, we
never meet twice the same node. Therefore, after a finite number
of steps, the algorithm gives us a solution to problem (P). If
it is not the case, we can conclude that the constraints (C-1),
(C-2) and (C-3) are incompatibles.

1-14 NUMBER OF NODES STORED IN THE MAIN CORE OF THE COMPUTER

Most of the branch and bound methods have a big disadvantage:
The number of nodes to be stored is increasing very rapidly (about
2^n). For large-scale mathematical programming problem, one has to
use secondary storage, in order to store all the informations
relative to each node. The disadvantage associated with the usage
of secondary storage is that the execution time increases very rapidly.

In order not to use secondary storage (and therefore in order
to reduce the execution time), one has to limit the number of nodes
to be stored in the main memory. A careful look at the different
states of the arborescence using the BBB method, indicates that
we store at the most an accepted node at each level. Using the fact
that we store the root but not the terminal node, the number of
nodes stored is equal to N, where N is the number of integer variables
of the problem (P). We can therefore expect to solve large-scale
integer nonlinear programming problem in a reasonable time.

Note

Abadie,Akoka and Dayan (19)(20)have developed two other arborescent methods (called The Bounded Method" and the" Modified Bounded Method".) These methods were tested on the problems used in II-1. As expected, the BBB method converged more rapidly . (The CPU times were significantly lower for BBB). Besides, the number of nodes of the arborescence for the "Bounded Method" and the "modified bounded method" was significantly bigger than for BBB.

I-14 Example

Let's apply BBB to the following problem.

(P) $\cdot$ min ϕ (X)=$(5X_1 - 13)^2 + 15(130 \, X_1 - 100 \, X_2 - 23)^2 + 30(20X_1 + 11X_2 -$

$-20X_3 - \quad)^2$

$\left\{\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right.$ 5 $0 \leq X_j \leq 6,$ $J\Sigma J= \{1,2,3\}$ (1)

          X_j integer, $J\Sigma E=J$ (2)

ϕ(X) is convex and differentiable. The convex space is R^3. By solving

the continuous nonlinear programming problem (i.e. (1) without constraint

(2)), we obtain:

$\hat{X}(\phi)$ = (2.6; 3.15; 3.75), $\phi[\hat{X}(\phi)]$ = 0. The solution of (P) is:

  $\hat{X}$ = (1;1;1), $\phi(\hat{X})$=829

In the following pages, we   present the results.

Column 1: T=(B_1X_B) is the node examined in the arborescence. It

          corresponds to the number of the iteration.

Column 2: Set B. It is obtained by applying one of the five criteria

          described in (I   ). In this case, we urged criterion #2.
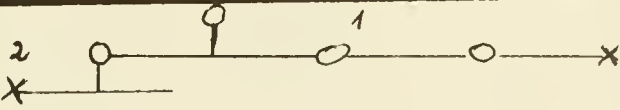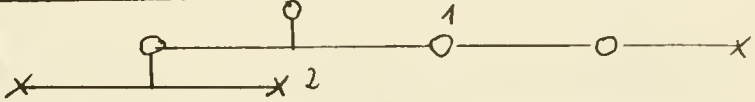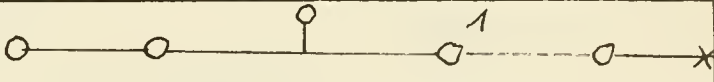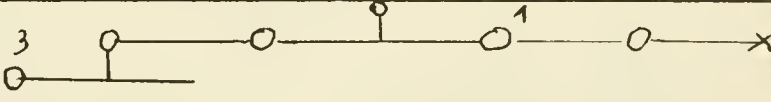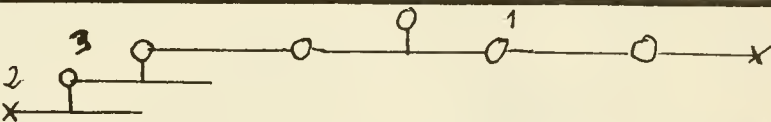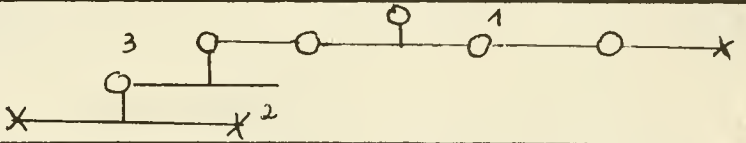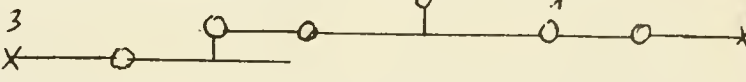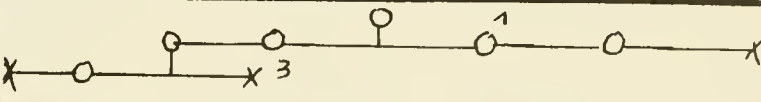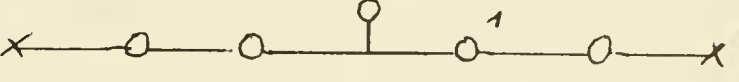
Column 3, The values of X_1,X_2,X_3 for the optimal solution of P(T). We
4,5

          give only the integer values. The asterisk means that X_j

          has been fixed and is integer (i.e. j = B).

Column 6: $\hat{\phi}$ = the value of the objective function. The optimal

value is underlined. We give only the values that improve

the objective function.

Column 7: The state of the arborescence.

Column 8: Value of the level t_m which is equal to the number of integer

variables.

T	B	$\hat{x}(T)$			$\hat{\phi}_T$	State of the Arborescence	t_m
		x_1	x_2	x_3			
0	∅	2.6	3.15	3.75	0		0
1	1	3*	3.67	4.51	4		1
2	1, 3	3		5*			2
3	1,3,2	3	4*	5	17419		3
4	1,3,2	3	3*	5			3
5	1,3	3		6*			2
6	1,3	3		4*			2
7	1,3,2	3	4*	4			3
8	1,3,2	3	3*	4			3
9	1,3	3		3*			2
10	1	4*					1
11	1,2	4	5*	6	934		2
12	1,2	4	4*	6			2
13	1	5*					1
14	1	2*					1

r	B	$\hat{x}(T)$			$\hat{\phi}_T$	State of the Arborescence	t_m
		x_1	x_2	x_3			
15	1, 2	2	2*				2
16	1, 2	2	3*				2
17	1	1*					1
18	1, 3	1		1*			2
19	1, 3, 2	1	1*	1	<u>829</u>		3
20	1, 3, 2	1	2*	1			3
21	1, 3	1		0*			2
22	1, 3	1		2*			2
23	1	0*					1

SOLUTION   T = 19

The final graph for the problem is given below.

$\overset{j}{\underset{k}{\textcircled{S_i}}}$ means that the node i is under investigation; variable x_j is being fixed at the value k.
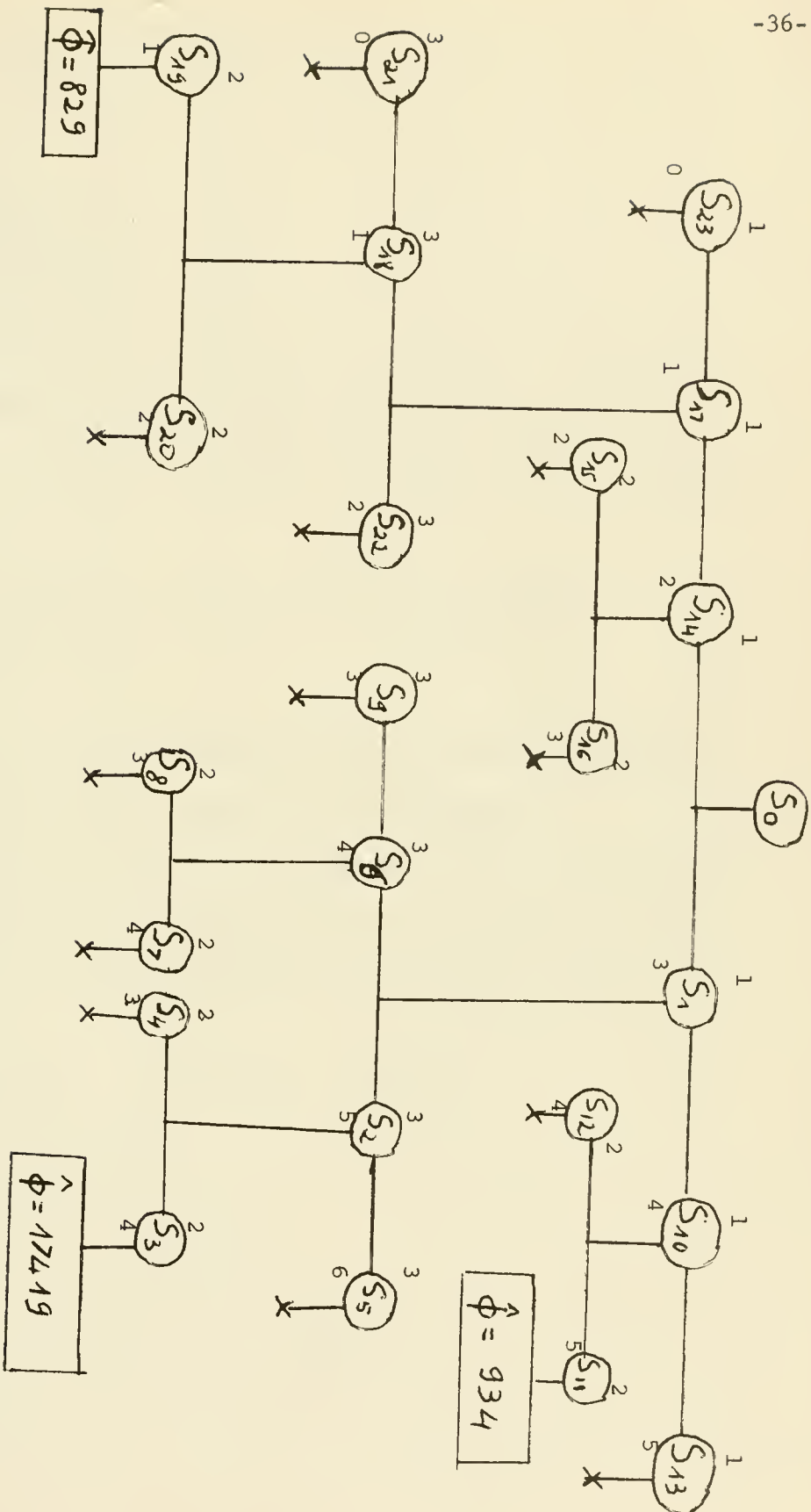
II. Numerical Computation

II.1 Introduction

The BBB method and the criteria proposed in chapter I have been programmed in FORTRAN IV. They have been tested on a CDC 6600 using the SCOPE system. 10 problems of small size (between 3 and 10 variables) have been used.

4 Problems are convex. 5 other are non-convex and there is one problem with Boolean variables.

Even though the problems may seem easy, nevertheless, they allow us to see all the difficulties related to the arborescence.
The CPU times necessary to solve the problems are very slow. Therefore, we may be able to solve a large-scale problems in a reasonable time. In the following page, we present the results for 9 problems. We use the following notations:

ϕ = optimal value of the objective function

CP = CPU time, in seconds

NTE = total number of evaluations

NTO = total number of optimizations

NSPA = number of nodes temporary accepted

NRFT = number of refusal using $\widetilde{\phi}$ (see I-7-B)

	CRITERION #1	CRITERION #2	CRITERION #3	CRITERION #4	CRITERION #5
PROBLEM N° 1	φ = 829 CP = 0.345 NTE = 16 NTO = 38 NSPA = 13 NRFT = 0	φ = 829 CP = 0.272 NTE = 8 NTO = 24 NSPA = 8 NRFT = 0	φ = 829 CP = 0.404 NTE = 12 NTO = 40 NSPA = 14 NRFT = 0	φ = 829 CP = 0.423 NTE = 14 NTO = 43 NSPA = 15 NRFT = 0	φ = 829 CP = 0.282 NTE = 8 NTO = 27 NSPA = 9 NRFT = 0
PROBLEM N° 2	φ = 728 CP = 7.165 NTE = 48 NTO = 324 NSPA = 109 NRFT = 0	φ = 728 CP = 8.195 NTE = 80 NTO = 386 NSPA = 122 NRFT = 0	φ = 728 CP = 7.812 NTE = 48 NTO = 348 NSPA = 117 NRFT = 0	φ = 728 CP = 5.298 NTE = 46 NTO = 251 NSPA = 84 NRFT = 0	φ = 728 CP = 7.268 NTE = 50 NTO = 310 NSPA = 104 NRFT = 0
PROBLEM N° 3	φ = -44 CP = 0.697 NTE = 2 NTO = 9 NSPA = 4 NRFT = 0	φ = -44 CP = 1.101 NTE = 2 NTO = 8 NSPA = 4 NRFT = 3	φ = -44 CP = 0.508 NTE = 4 NTO = 11 NSPA = 5 NRFT = 3	φ = -44 CP = 0.898 NTE = 2 NTO = 8 NSPA = 4 NRFT = 1	φ = -44 CP = 1.630 NTE = 6 NTO = 23 NSPA = 9 NRFT = 1
PROBLEM N° 4	φ = 38 CP = 0.191 NTE = 4 NTO = 17 NSPA = 8 NRFT = 0	φ = 38 CP = 0.127 NTE = 4 NTO = 11 NSPA = 4 NRFT = 0	φ = 38 CP = 0.130 NTE = 4 NTO = 11 NSPA = 4 NRFT = 0	φ = 38 CP = 0.086 NTE = 4 NTO = 10 NSPA = 4 NRFT = 1	φ = 38 CP = 0.151 NTE = 4 NTO = 16 NSPA = 5 NRFT = 0
PROBLEM N° 5	φ = -28.2857 CP = 0.130 NTE = 1 NTO = 2 NSPA = 1 NRFT = 0	φ = -26.2857 CP = 0.129 NTE = 1 NTO = 2 NSPA = 1 NRFT = 0	φ = -28.2857 CP = 0.129 NTE = 1 NTO = 2 NSPA = 1 NRFT = 0	φ = -28.2857 CP = 0.135 NTE = 1 NTO = 2 NSPA = 1 NRFT = 0	φ = -26.2857 CP = 0.129 NTE = 1 NTO = 2 NSPA = 1 NRFT = 0
PROBLEM N° 6	φ = 30.4552,93 CP = 0.130 NTE = 4 NTO = 8 NSPA = 3 NRFT = 0	φ = 30.4552,93 CP = 0.172 NTE = 8 NTO = 15 NSPA = 5 NRFT = 0	φ = 30.4552,83 CP = 0.140 NTE = 4 NTO = 8 NSPA = 3 NRFT = 0	φ = 30.4552,93 CP = 0.129 NTE = 4 NTO = 8 NSPA = 3 NRFT = 0	φ = 30.4552,9 CP = 0.172 NTE = 8 NTO = 15 NSPA = 5 NRFT = 0
PROBLEM N° 7	φ = 20 CP = 0.600 NTE = 2 NTO = 8 NSPA = 8 NRFT = 0	φ = 20 CP = 0.471 NTE = 2 NTO = 7 NSPA = 5 NRFT = 0	φ = 20 CP = 0.613 NTE = 2 NTO = 8 NSPA = 4 NRFT = 0	φ = 20 CP = 0.641 NTE = 2 NTO = 6 NSPA = 4 NRFT = 0	φ = 20 CP = 0.509 NTE = 2 NTO = 7 NSPA = 5 NRFT = 0
PROBLEM N° 8	φ = -14.10 CP = 0.151 NTE = 3 NTO = 8 NSPA = 3 NRFT = 0	φ = -14.10 CP = 0.259 NTE = 3 NTO = 5 NSPA = 2 NRFT = 0	φ = -14.10 CP = 0.138 NTE = 1 NTO = 3 NSPA = 1 NRFT = 0	φ = -14.10 CP = 0.172 NTE = 3 NTO = 6 NSPA = 3 NRFT = 0	φ = -14.10 CP = 0.130 NTE = 1 NTO = 3 NSPA = 1 NRFT = 0
PROBLEM N° 9	φ = -72 CP = 0.046 NTE = 1 NTO = 1 NSPA = 0 NRFT = 0	φ = -72 CP = 0.046 NTE = 1 NTO = 1 NSPA = 0 NRFT = 0	φ = -72 CP = 0.048 NTE = 1 NTO = 1 NSPA = 0 NRFT = 0	φ = -72 CP = 0.046 NTE = 1 NTO = 1 NSPA = 0 NRFT = 0	φ = -72 CP = 0.046 NTE = 1 NTO = 1 NSPA = 0 NRFT = 0

II.3 Rank of the Criteria

Let's describe two methods of ranking the criteria. The first method, due to COLVILLE use the mean and the variance of the time necessary to solve problem i by method j. The second method, due to ABADIE, use the best time obtained for the problem.

II.1.1   COLVILLE'S METHOD

Let t_{ij} be the time necessary to solve problem i using criterion j. Define $\bar{t}_{ij}$ = mean of t_{ij}

$$\sigma_{t_{ij}} = \text{standard error of } t_{ij}$$

For each criterion   and for each problem, we compute the following quantity:

$$q_{ij} = \frac{-t_{ij} + \bar{t}_{ij}}{\sigma_{t_{ij}}}$$

If n is equal to the number of problems solved using criterion j, the ranking of the criterion j is given by:

$$Q_j = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} q_{ij}$$

II.1.2   ABADIE'S METHOD

Let t_{ij} be the time necessary for criterion j to solve problem.
Let $\min\limits_{j} (t_{ij})$ be the best time obtained for problem i.
We compute the following quantity:

$$h_{ij} = \frac{t_{ij}}{\min\limits_{j} (t_{ij})}$$

If n represents the number of problems solved, we obtain a

a rank for the criterion j using the following forumula

$$H_i = \frac{1}{n} \sum_{i=1}^{n} \frac{t_{ij}}{\min_j (t_{ij})}$$

If we apply both methods to Table 1 representing the CPU times
necessary to solve each problem using the five criteria, we obtain a ranking
for each criteria. The values for each criterion is given in Table 2.

PROBLEMS

		I	II	III	IV	V	VI	VII	VIII
	1	0.345	7.165	0.697	0.191	0.130	0.130	0.600	0.151
CRITERIA	2	0.272	8.195	1.101	0.127	0.129	0.172	0.471	0.259
	3	0.404	7.912	0.508	0.130	0.129	0.140	0.613	0.138
	4	0.423	5.296	0.698	0.086	0.135	0.129	0.641	0.172
	5	0.282	7.268	1.630	0.151	0.129	0.172	0.509	0.130

TABLE 1

TABLE 2	Number of Problems	ABADIE'S METHOD	COLVILLE'S METHOD
CRITERION 1	8	1.33315	0.079349E-2
CRITERION 2	8	1.43964	-0.2017656
CRITERION 3	8	1.2424	0.139656
CRITERION 4	8	1.20746	0.225989
CRITERION 4	8	1.47345	-0.171816

The ranking given by ABADIE AND COLVILLE'S methods is given in

<u>Table</u> 3

CRITERIA	ABADIE'S METHOD
4	1.20746
3	1.2424
1	1.33315
2	1.43964
5	1.47345

CRITERIA	COLVILLE'S METHOD
4	0.225989
3	0.139656
1	0.079349E-2
5	-0.171816
2	-0.2017656

<u>TABLE   3</u>

The best criterion is 4 (i.e. put in the list the problem corresponding

to max $(A_j f_j,\ B_j(1-f_j)$

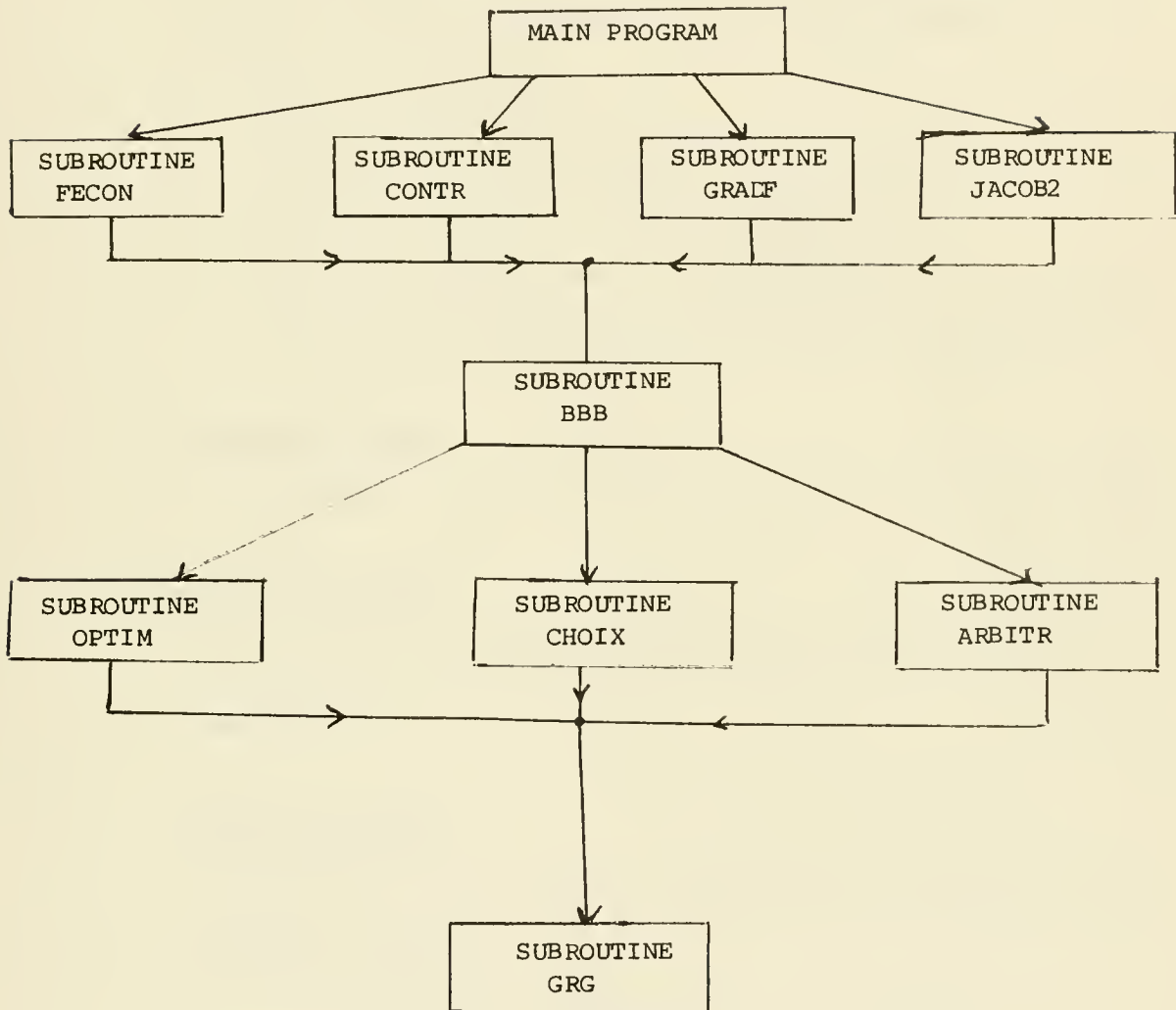As an indication, we give below another method to order the criteria.

Let x_{ij} be the total number of optimizations needed by criteria j to solve

problem i. Let $\min_{j}(x_{ij})$ be the smallest number of optimizations for

problem i. The following quantity gives us an order for each criteria:

$$M_i = \frac{1}{n} \sum_{i=1}^{n} \frac{x_{ij}}{\min_{j}(x_{ij})}$$

We feel that this quantity is useful when one has to do with linear

programming.

II.3   Presentation of the Computer Code

The flowchart of the code is:

```
                          ┌─────────────────┐
                          │  MAIN PROGRAM   │
                          └─────────────────┘

┌────────────┐   ┌────────────┐   ┌────────────┐   ┌────────────┐
│ SUBROUTINE │   │ SUBROUTINE │   │ SUBROUTINE │   │ SUBROUTINE │
│   FECON    │   │   CONTR    │   │   GRADF    │   │   JACOB2   │
└────────────┘   └────────────┘   └────────────┘   └────────────┘


                          ┌─────────────────┐
                          │   SUBROUTINE    │
                          │       BBB       │
                          └─────────────────┘


┌────────────┐   ┌────────────┐   ┌────────────┐
│ SUBROUTINE │   │ SUBROUTINE │   │ SUBROUTINE │
│   OPTIM    │   │   CHOIX    │   │   ARBITR   │
└────────────┘   └────────────┘   └────────────┘


                          ┌─────────────────┐
                          │   SUBROUTINE    │
                          │       GRG       │
                          └─────────────────┘
```

Let us now detail each block of the flowchart.

A. The Main Program

In the main program, the user has to initialize the problem. The user indicates the following parameters

- number of variables

- upper bound and lower bound for each variable

- number of equalities in the constraints

- number of inequalities in the constraints

- characteristic of the problem (convex or non-convex)

B. Subroutine Fecon

In this subroutine, the user has to give the objective function.

C. Subroutine Contr

The user has to specify in this subroutine, the set of constraints.

D. Subroutine Gradf

The user must indicate in this subroutine, the gradient of the objective function:

$$DfI \ (I) = \frac{\partial \phi}{\partial X(1)}$$

E. Subroutine Jacob2

The user must indicate the Jacobian of the constraints

$$C(I,J) = \frac{\partial \ VC(x)}{\partial \ x_j}$$

F. <u>Subroutine BBB</u>

This subroutine is described in details in Chapter 3. BBB

calls OPTIM which leads to the solution of the continuous

nonlinear programming problem. It calls also CHOIX and ARBITR

in order to use the five criteria defined in (II.1).

G. <u>Subroutine Optim</u>

Called by BBB, optim considers the variables not yet integer

and solve the continupus nonlinear programming problem using

GRG. It searches the fortuitous integer variables (see I.10).

H. <u>Subrouting Choix</u>

This subroutine allows to choose the "separation variable".

(see I.11).

It uses the five criteria defined in (I-11).

I. <u>Subroutine Arbitr</u>

It allows us to determine whether the optimization by GRG

leads to an optimal solution.

J. <u>Subroutine GRG</u>

It is the code used by ABADIE and ranked first by COLVILLE.

It allows to solve continuous nonlinear programming problems (4).

BIBLIOGRAPHY

(1) - J. ABADIE (Ed.)   INTEGER AND NON-LINEAR PROGRAMMING
NORTH-HOLLAND PUBLISHING COMPANY and
AMERICAN ELSEVIER PUBLISHING COMPANY (1970)

(2) - J. ABADIE   NON-LINEAR PROGRAMMING
NORTH-HOLLAND PUBLISHING COMPANY (1967)

(3) - J. ABADIE   PROBLEMES D'OPTIMISATION
Tome I et II
INSTITUT BLAISE PASCAL - PARIS -(1976)

(4) - J. ABADIE - J. GUIGOU   GRADIENT REDUIT GENERALISE
NOTE E.D.F. HI 069/02 -(15 AVRIL 1969)

(5) - J. ABADIE   UNE METHODE ARBORESCENTE POUR LES PROBLEMES
NON-LINEAIRES PARTIELLEMENT DISCRETS
RIRO - 3ème année - V-3 - (1969)

(6) - J. ABADIE   SOLUTION DES QUESTIONS DE DEGENERESCENCE
DANS LA METHODE GRG -
NOTE E.D.F. HI 143/00 - (25 SEPTEMBRE 1968)

(7) - J. ABADIE   UNE METHODE DE RESOLUTION DES PROGRAMMES NON-
LINEAIRES PARTIELLEMENT DISCRETS SANS HYPOTHESE
DE CONVEXITE - RIRO lère année V-1 (1971)

(8) - J. BRACKEN,G.P.Mc CORMICK SELECTED APPLICATIONS OF NON-LINEAR PROGRAMMING
John WILEY and SONS Inc. - (1968)

(9) - E.M.L. BEALE (Ed.)   APPLICATIONS OF MATHEMATICAL PROGRAMMING TECHNIQUES
THE ENGLISH UNIVERSITIES PRESS Ltd. (1970)

(10) - A.R. COLVILLE   A COMPARATIVE STUDY ON NON-LINEAR CODES
IBM/N.Y. SCIENTIFIC CENTER REPORT N° 320.2949
(1969)

(11) - L.C.W. DIXON   NON-LINEAR OPTIMISATION
THE ENGLISH UNIVERSITIES PRESS Ltd. (1972)

(12) - S.I. GASS   LINEAR PROGRAMMING (third edition)
Mc GRAW-HILL BOOK COMPANY

(13) - P.L. HAMMER - S.RUDEANU   METHODES BOOLEENNES EN RECHERCHE OPERATIONNELLE
DUNOD PARIS (1970)

(14) - D.M.HIMMELBLAU   APPLIED NON-LINEAR PROGRAMMING
Mc GRAW HILL BOOK COMPANY (1972)

(15) - J.C.T. MAO   QUANTITATIVE ANALYSIS OF FINANCIAL DECISIONS
THE MAC MILLAN COMPANY
COLLIER-MAC MILLAN LIMITED - LONDON (1969)

(16) - J. GUIGOU   PRESENTATION ET UTILISATION DU CODE GRG
NOTE E.D.F. HI 102/02 (9 JUIN 1969)

(17) - O.L. MANGASARIAN   NON-LINEAR PROGRAMMING
Mc GRAW-HILL SERIES IN SYSTEM SCIENCE
Mc GRAW-HILL COMPANY (1969)

(18) - W.I.ZANGWILL   NON-LINEAR PROGRAMMING : A UNIFIED APPROACH
PRENTICE HALL INTERNATIONAL SERIES IN MANAGEMENT
PRENTICE HALL INC. (1969)

(19) J. AKOKA   METHODES ARBORESCENTES DE RESOLUTION
DES PROGRAMMES NON LINEARES TOTALEMENT
OU PARTIELLEMENT DISCRETS DOCTORAT 3eme
CYCLE - UNIVERSITE DE PARIS VI - JUNE 1975.

(20) J. ABADIE, J. AKOKA,   METHODES ARBORESCENTES EN PROGRAMMATION
      H. DAYAN   NONLINEAIRE ENTIERE. (to appear in
R.A. R.)